# CLAIMS

1. A method for rendering soft shadows in a frame representing a three-dimensional scene, which is defined by a list of polygons and comprises a plurality of three-dimensional objects, said method comprising the steps of:

from a light source's point of view:

- determining edges casting shadows from said list of polygons;
- computing shadow volumes and computing soft shadow edges from said edges casting shadows;
- creating up to six squared empty textures of the same size, each of them becoming a face of a cubemap representing a cube centered at the light source's point of view, said faces of said cubemap being aligned with orthogonal major axes with the light source's position as origin;
- rendering into at least one face of said cubemap an appropriate part of said scene with full brightness color and rendering a representation of soft shadows into said at least one face of said cubemap using said edges casting shadows and said soft shadow edges computed;

from a point of view of a viewer viewing said scene:

- rendering a visible part of the scene into a z-buffer with colors, lighting and texture information disabled for generating depth information;
- rendering said shadow volumes into a stencil buffer in combination with said z-buffer information; and
- in a single pass, rendering said scene with colors, lighting and texture information enabled and applying said cubemap to said scene for rendering said representation of said soft shadows into said scene using a texture coordinate generation while performing a stencil test operation for preventing the scene to be drawn in shadowed areas, to produce a soft shadowed image.

2. The method according to claim 1, wherein the step of rendering a representation of soft shadows into said at least one face of said cubemap comprises:

interpolating color or alpha values between vertex color or alpha values of said soft shadow edges and vertex color or alpha values of said edges casting shadows.

3. The method according to claim 2, wherein the step of computing soft edges comprises:

for each edge casting shadows creating an additional polygon formed by four vertices and setting two vertices of said additional polygon to a respective endpoint of said edge casting shadows, the color or alpha value of each of said two vertices being set to full ambient darkness;

computing, from said two vertices set to full ambient darkness, two remaining vertices at a predetermined distance from said two vertices set to full ambient darkness; and

setting the color or alpha value of each said remaining two vertices being set to full ambient brightness.

4. The method according to the preceding claim, wherein the step of computing said two remaining vertices comprises:

computing a vector given by a cross product between a normalized vector along said edge casting shadows and a normal onto a polygon surface associated to said edge casting shadows and setting the remaining two vertices of said additional polygon to points at a predetermined distance from said two vertices in a direction given by said computed vector.

5. The method according to the preceding claim, wherein

the position of the remaining two vertices is further modified by computing interpolated vectors by respectively interpolating said computed vector with another vector correspondingly computed for an adjacent edge casting shadows and

the remaining two vertices of said additional polygon are respectively set to points at a predetermined distance from said two vertices in a direction given by said interpolated vector.

6. The method according to claim 1, further comprising the steps of

determining which sides of said cubemap requires an update based information on at least one of the viewer's position in the scene, the viewer's direction of view, the viewer's field of view, the light source's position and the position of any object moving in said scene; and

rendering said representation of said soft shadow edges into each side of said cubemap that has been determined requiring an update.

7. The method according to claim 1, wherein the step of rendering said shadow volume polygons into the stencil buffer comprises:

clearing the stencil buffer to a nil value;

performing a per-pixel stencil operation

- for increasing a value present in the stencil buffer for front facing shadow volume polygons, if the depth value of the shadow volume polygon at the pixel of interest is less than the depth value stored in the z-buffer at the pixel of interest; and
- decreasing a value present in the stencil buffer for back facing shadow volume polygons, if the depth value of the shadow volume polygon at the pixel of interest is less than the depth value stored in the z-buffer at the pixel of interest.

8. The method according to the preceding claim, wherein the step of applying said cubemap while performing a stencil test operation comprises:

performing a per-pixel test against values stored in the stencil buffer and preventing rendering a fragment of said scene at the pixel of interest if the value stored in the stencil buffer at the pixel of interest is different from the nil value;

accessing said cubemap via vectors given by 3D texture coordinates, where the greatest magnitude component is used to select a face of said cubemap and the other two components are used to select a texel from said face; and

performing a texture coordinate generation for specifying a texture coordinate and selecting a pixel of an appropriate cubemap face.

9. The method according to claim 1, wherein the steps of, from the light source's point of view, determining edges casting shadows, computing shadow volumes and soft shadow edges, creating said cubemap and rendering into at least one face of said cubemap an appropriate part of said scene and a representation of soft shadows are repeated for every light source of the scene.

10. The method according to claim 1, further comprising the steps of

storing information on characteristic points of shadows cast in a first frame;

performing the steps according to any of the preceding claims for a subsequent frame of an animation of said three-dimensional scene; and

computing information on corresponding characteristic points of shadows cast in said subsequent frame; wherein

soft shadows in said subsequent frame are only re-rendered if a difference value computed from corresponding characteristic points of the two frames exceeds a predetermined threshold.

11. A computer program product for rendering soft shadows in a frame representing a three-dimensional scene, which is defined by a list of polygons and comprises a plurality of three-dimensional objects, comprising:

program code means for performing, from a light source's point of view, the steps of:

- determining edges casting shadows from said list of polygons;
- computing shadow volumes and computing soft shadow edges from said edges casting shadows;
- creating up to six squared empty textures of the same size, each of them becoming a face of a cubemap representing a cube centered at the light source's point of view, said faces of said cubemap being aligned with orthogonal major axes with the light source's position as origin;

- rendering into at least one face of said cubemap an appropriate part of said scene with full brightness color and rendering a representation of soft shadows into said at least one face of said cubemap using said edges casting shadows and said soft shadow edges computed;

program code means for performing, from a point of view of a viewer viewing said scene, the steps of:

- rendering a visible part of the scene into a z-buffer with colors, lighting and texture information disabled for generating depth information;
- rendering said shadow volumes into a stencil buffer in combination with said z-buffer information; and
- in a single pass, rendering said scene with colors, lighting and texture information enabled and applying said cubemap to said scene for rendering said representation of said soft shadows into said scene using a texture coordinate generation while performing a stencil test operation for preventing the scene to be drawn in shadowed areas, to produce a soft shadowed image.

12. The computer program product according to claim 11, wherein a program code means for rendering a representation of soft shadows into said at least one face of said cubemap comprises:

program code means for interpolating color or alpha values between vertex color or alpha values of said soft shadow edges and vertex color or alpha values of said edges casting shadows.

13. The computer program product according to claim 12, wherein a program code means for computing soft edges comprises:

program code means for creating, for each edge casting shadows, an additional polygon formed by four vertices and setting two vertices of said additional polygon to a respective endpoint of said edge casting shadows, the color or alpha value of each of said two vertices being set to full ambient darkness;

program code means for computing, from said two vertices set to full ambient darkness, two remaining vertices at a predetermined distance from said two vertices set to full ambient darkness; and

program code means for setting the color or alpha value of each said remaining two vertices being set to full ambient brightness.

14. The computer program product according to the preceding claim, wherein the program code means for computing said two remaining vertices comprises:

program code means for computing a vector given by a cross product between a normalized vector along said edge casting shadows and a normal onto a polygon surface associated to said edge casting shadows and setting the remaining two vertices

of said additional polygon to points at a predetermined distance from said two vertices in a direction given by said computed vector.

15. The computer program product according to the preceding claim, further comprising:

program code means for modifying the position of the remaining two vertices by computing interpolated vectors by respectively interpolating said computed vector with another vector correspondingly computed for an adjacent edge casting shadows and

program code means for setting the remaining two vertices of said additional polygon respectively to points at a predetermined distance from said two vertices in a direction given by said interpolated vector.

16. The computer program product according to claim 11, further comprising:

program code means for determining which sides of said cubemap requires an update based information on at least one of the viewer's position in the scene, the viewer's direction of view, the viewer's field of view, the light source's position and the position of any object moving in said scene; and

program code means for rendering said representation of said soft shadow edges into each side of said cubemap that has been determined requiring an update.

17. The computer program product according to claim 11, wherein the program code means for rendering said shadow volume polygons into the stencil buffer comprises:

program code means for clearing the stencil buffer to a nil value;

program code means for performing a per-pixel stencil operation

- for increasing a value present in the stencil buffer for front facing shadow volume polygons, if the depth value of the shadow volume polygon at the pixel of interest is less than the depth value stored in the z-buffer at the pixel of interest; and
- decreasing a value present in the stencil buffer for back facing shadow volume polygons, if the depth value of the shadow volume polygon at the pixel of interest is less than the depth value stored in the z-buffer at the pixel of interest.

18. The computer program product according to the preceding claim, wherein a program code means for applying said cubemap while performing a stencil test operation comprises:

program code means for performing a per-pixel test against values stored in the stencil buffer and preventing rendering a fragment of said scene at the pixel of interest if the value stored in the stencil buffer at the pixel of interest is different from the nil value;

program code means for accessing said cubemap via vectors given by 3D texture coordinates, where the greatest magnitude component is used to select a face of said cubemap and the other two components are used to select a texel from said face; and

program code means for performing a texture coordinate generation for specifying a texture coordinate and selecting a pixel of an appropriate cubemap face.

19. The computer program product according to claim 11, wherein program code means for, from the light source's point of view, determining edges casting shadows, computing shadow volumes and soft shadow edges, creating said cubemap and rendering into at least one face of said cubemap an appropriate part of said scene and a representation of soft shadows are repeatedly operated for every light source of the scene.

20. The computer program product according to claim 11, further comprising:

program code means for storing information on characteristic points of shadows cast in a first frame;

program code means for performing the steps according to claim 11 for a subsequent frame of an animation of said three-dimensional scene; and

program code means for computing information on corresponding characteristic points of shadows cast in said subsequent frame; and

program code means for re-rendering soft shadows in said subsequent frame only if a difference value computed from corresponding characteristic points of the two frames exceeds a predetermined threshold.

21. A system for rendering a three-dimensional scene comprising a plurality of three-dimensional objects formed by polygons and for rendering soft shadows in said scene:

a) a color buffer for storing color values;

b) a z-buffer for storing depth values;

c) a stencil buffer for storing stencil mask information;

d) texture memory for storing texture information;

e) a rasterizer for converting each of the polygons into color, depth and stencil values;

f) a first pixel engine in communication with the rasterizer and the texture memory for copying or drawing directly a portion of the color buffer into an appropriate texture representing a specific cubemap side;

g) a second pixel engine in communication with the rasterizer, the z-buffer and the stencil buffer for receiving said polygons and performing a depth compare operation and a stencil test operation based on the result of the depth compare operation; and

h) a third pixel engine in communication with the rasterizer, the z-buffer, the stencil buffer and the texture memory for receiving said polygons and performing a

depth compare operation, a stencil test operation and a texture coordinate generation operation, based on the result from the depth compare and stencil test, for transferring into the non-shadowed areas of the frame buffer the appropriate pixels of the appropriate side of the cubemap.

5

22. The system according to the preceding claim, further comprising an alpha buffer for rendering into the cubemap 8 bit alpha values instead of 24 bit RGB values or any other value stored in the color buffer.